VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Absolvování individuální odborné praxe
Individual Professional Practise in the Company

2010

Jiří Stařinský

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne …....

Jiří Stařinský

ABSTRAKT

Má bakalářská práce obsahuje zhodnocení průběhu individuální odborné praxe ve firmě ITA, s.r.o., kde jsem pracoval na pozici programátora a věnoval se udržování stávajících aplikací a vývoji nových aplikací. Popisuji firmu, moji funkci ve firmě a naznačuji postup při řešení jednotlivých úkolů, které mi byly zadávány.

Zhodnocuji uplatněné teoretické a praktické znalosti z bakalářského studia a také scházející znalosti.

V závěru práce prezentuji část mých dosažených výsledků, úspěchů a celkový přínos odborné praxe.

Klíčová slova:

ITA, DLPP, Strip Observer

ABSTRACT

My bachelor thesis contains the evaluation and rewiev of my individual professional practise in the ITA company, in where I worked at the position of programmer. During my practise there, I worked at maintance of existing applications and participate on developing new software. I am going to make an attempt to describe the company briefly, my working position in the company and I am going to release the process of solving single tasks, which were set to me.

There is going to be a conclusion of my theoretical and practical knowledges that I have gained during my study of bachelor degree and I will also try to release the missing part of knowledges.

At the end I am going to present a part of fulfilled issues, successes and a global advantages of my practise.

Keywords:

ITA, DLPP, Strip Observer

Poděkování

Na tomto místě bych rád poděkoval panu Ing. Pavlu Šimečkovi, Ph.D. a také panu Ing. Danieli Hajdukovi, Ph. D., jednatelům společnosti ITA, s. r. o., za možnost absolvovat odbornou praxi v jejich společnosti.

OBSAH

1 POPIS ODBORNÉHO ZAMĚŘENÍ FIRMY ITA, S. R. O., A POPIS PRACOVNÍHO ZAŘAZENÍ STUDENTA

Firma ITA, s. r. o., je soukromá společnost založená roku 1991 výzkumnými a vědeckými pracovníky Výzkumného ústavu strojírenského a metalurgického VÍTKOVICE.

Zabývá se zejména moderními technologiemi válcování za tepla i za studena. Také dodává know-how, algoritmy a programová řešení významným producentům válcovacích zařízení a technologií (Danieli, Conver Team, SMS Demag a další) a řeší konkrétní technické problémy a technologické inovace na válcovacích tratích (ArcelorMittal Ostrava, Severstal Čerepovec, Třinecké železárny Třinec, AMAG Ranshofen, ArcelorMittal Vanderbijlpark a jiné).

Mým zaměřením v této společnosti byla údržba a rozšiřování stávajících programových řešení. Dále jsem se v průběhu praxe od začátku podílel na vývoji nových softwarových řešení, zejména pro společnost Conver Team.

2 ÚKOLY ŘEŠENÉ V PRŮBĚHU PRAXE

Jak už bylo zmíněno výše, mým úkolem byla zejména údržba stávajících aplikací. Většina aplikací byla psaných v jazycích C/C++ za použití technologií MFC, nativního Win32 API a klientských kódů využívajících COM objekty. Proto, abych byl vůbec schopen pracovat s těmito jazyky a technologiemi, jsem se nejprve musel věnovat jejich studiu. Zdrojem informací mi byly zejména odborné knihy, články a různá komunitní fóra zabývající se vývojem aplikací na platformě Windows. Současně mi také velmi pomohlo studium programového kódu již napsaných modulů a aplikací společností ITA, s. r. o.

2.1 DLPP

Plným názvem Danieli Long Products Properties Prediction. Tento program umožňuje výpočet mechanických vlastností za tepla válcovaných ocelových tyčí a drátů pro různé konfigurace tratí a různé technologie válcování a ochlazování, dále pak výpočet rozvoje mikrostruktury během válcování. Program DLPP je vyvíjen společností ITA, s. r. o., již od roku 2005 pro významného zahraničního klienta Danieli. Primární činností této firmy je distribuování zařízení pro hutní průmysl. Současně tato společnost vlastní několik továren v Evropě a Asii.

Než jsem vůbec mohl program DLPP jakkoliv upravovat a rozšiřovat, musel jsem se nejprve seznámit s použitou terminologií v této oblasti, pochopit některé základní postupy výpočtů a v neposlední řadě naučit se obsluhovat stávající verzi produktu. Také jsem byl seznámen s potřebnými úpravami a opravami, které mi byly svěřeny.

Po této fázi jsem byl připraven začít studovat programátorskou dokumentaci a zdrojový kód programu a modulů, které tento program využívá. Kvůli rozsáhlosti programu a také díky mé prvotní nezkušenosti s architekturou MFC aplikací a takovýmito projekty mi tato část práce zabrala poměrně dlouhou dobu.

Jako první z mé činnosti na tomto projektu bylo odstranění stávajících chyb programu, kvůli kterým se program stával nestabilním. Při odstraňování většiny programových chyb je klíčové umět chybu správně reprodukovat. K tomu je ovšem potřeba mít k dispozici dostatek informací o tom, jak a kdy chyba vznikala. Po úspěšném reprodukování chyb jsem mohl přistoupit k jejich lokalizaci a následnému odstranění.

Pro ladění programu jsem používal jak vestavěné ladicí nástroje ve vývojovém prostředí MS Visual C++ 6.0, tak i nástroje třetích stran (WinDbg). Díky těmto nástrojům se mi chyby podařilo identifikovat a následně také v programu nalézt a odstranit.

Po odstranění většiny detekovaných chyb jsem mohl přistoupit k samotným požadavkům na úpravu a rozšíření stávající aplikace. V první verzi mnou editovaného programu se spíše jednalo jen o úpravy v uživatelském rozhraní aplikace. Ačkoliv se dá říci, že se spíše jednalo o banální úpravy, jejich realizace mi zabrala také dost času. Paradoxně pochopit architekturu MFC aplikací a celkového využití tohoto frameworku mi trvalo déle než samotné odstranění hlášených chyb. Nicméně postupem času se mi povedlo celkem zdatně zvládnout tuto komplexní knihovnu tříd.

Další úpravy programu byly spíše zaměřeny na rozšíření jeho stávající funkcionality. Mezi ty zajímavější úpravy bych zařadil úpravy týkající se některých výpočetních algoritmů. Tyto algoritmy byly upraveny případně zcela změněny na základě matematických vzorců dodaných přímo firmou Danieli. Díky těmto a jim podobným úpravám jsem měl možnost podílet se na implementaci těch nejdůležitějších částí programu. Mimo jiné mi tato skutečnost umožnila nahlédnout, jakým způsobem lze realizovat výpočty pomocí metody konečných prvků. Za zmínku také stojí rozšíření programu o nové typy zařízení, doplnění jejich grafické reprezentace do uživatelského rozhraní aplikace, implementace výpočtových částí charakterizujících dané zařízení a také řešení persistentního uložení dat. Pro serializaci resp. deserializaci původní autor zvolil soubory ve formátu XML. Pro práci s XML byl zvolen framework MSXML, což je produkt firmy Microsoft, přičemž je dodáván jako tzv. COM komponenta.

Mezi jedny z posledních úprav, které jsem na projektu prováděl, bylo vytvoření demo aplikace včetně instalátoru v prostředí MS Visual C++ 2005. V podstatě se jednalo jen o drobné úpravy kódu, které znemožnily využít program plnohodnotně.

2.2 Strip Observer

V druhé polovině své odborné praxe jsem se podílel na vývoji aplikace s názvem Strip Observer, která slouží k provádění teplotních a metalurgických výpočtů na základě informací získaných z databáze technologických parametrů procesu válcování pásů za tepla. Data do databáze exportuje automaticky řídicí systém válcovací trati. Úkolem programu Strip Observer je pak umožnit uživateli prohlížení těchto záznamů a následného provedení výše zmíněných výpočtů včetně grafické prezentace výsledků.

Vzhledem k důvěře a prostoru, jaký mi byl vedením firmy svěřen, jsem měl jedinečnou možnost podílet se na vývoji této aplikace od samého počátku zcela sám.

Při návrhu aplikace jsem čerpal zejména z rad svých kolegů. Do jisté míry mi pomohl i fakt, že cílem aplikace Strip Observer bylo počítat shodné charakteristiky jako v programu DLPP. Tudíž k pochopení algoritmů výpočtů jsem nemusel věnovat tolik času.

Jelikož se ve firmě využívá pro tvorbu softwaru iterativní vývoj, bylo možné v jednotlivých iteracích zjistit poměrně včas, zda mnou vytvořený návrh a implementace byl dostačující nebo naopak. Případné chyby tedy bylo možné odstranit a navrhnout nové řešení. Rovněž bylo možné relativně pružně reagovat na změnu určitých požadavků klienta.

Na vývoj aplikace bylo vyhrazeno celkem 8 měsíců rozdělených do vhodných časových úseků. V tomto dokumentu budu popisovat vývoj zejména v první iteraci. Požadavkem na produkt po první iteraci bylo především, aby byl schopen počítat potřebné charakteristiky.

Před samotným zahájením realizace mi vedení společnosti předalo požadavky zákazníka na systém. Po zdárném pochopení všech požadavků jsem mohl přistoupit k samotné analýze a návrhu. V průběhu analýzy jsem se nejprve snažil stanovit si scénáře, na základě kterých bylo později možné vytvořit případy užití.

Po návrhu případů užití jsem se rozhodl zvolit vhodnou architekturu aplikace. Rozhodujícím faktorem při výběru architektury se mi stala skutečnost, že v době vyvíjení aplikace Strip Observer zadavatel sám vyvíjel systém, který by exportoval data do databáze. Tudíž bez znalosti formátu exportovaných dat, typu databázového systému a potřebě vytvořit alespoň částečně fungující programový systém jsem se rozhodl pro použití architektury založené na myšlence tzv. vícevrstvé architektury. Přičemž pro realizaci datové vrstvy byla zvolena upravená verze návrhového vzoru DAO. Ve zkratce lze říci, že

návrhový vzor DAO obsahuje objekty, které implementují předem definované jednotné rozhraní pro komunikaci s datovým zdrojem. Díky takto definovanému rozhraní může aplikace přistupovat k datům uložených v databázi, souboru nebo jakéhokoliv jiného zdroje, aniž by musela znát přesný formát uložených dat.

Využitím tohoto vzoru se docela zdárně podařilo odstínit závislost na konkrétním datovém zdroji.

Charakteristickou vlastností této architektury je, že jednotlivé její části mohou běžet odděleně na různých strojích. Základ tvoří tři vrstvy: prezentační (frontend) – tato vrstva je zodpovědná za zobrazování výsledků a interakce s uživatelem pomocí uživatelského rozhraní, dále pak aplikační vrstva – obsahuje veškerou logiku aplikace a konečně datová vrstva (backend), zodpovědná za komunikaci s datovým zdrojem.

Navíc toto rozdělení do vrstev si s sebou přináší další řadu výhod. Mezi jednu z nejvýznamnějších patří možnost úpravy jednotlivých vrstev nezávisle na ostatních. Dále pak možnost mít vedle sebe několik různých prezentačních klientských programů.

Jelikož aplikace Strip Observer byla pouze standardní desktopovou aplikací, vzal jsem si z tohoto konceptu pouze myšlenku členění souvisejících částí do vrstev.

V další fázi návrhu aplikace bylo potřeba rozvrhnout vhodnou reprezentaci modelovaných problémů pomocí tříd. Zde jsem si na základě terminologie související s danou tématikou a diskuze s vedením, které z atributů budou dodány při importu dat, vytvořil jejich návrhy a definoval závislosti mezi nimi. Celý model jsem se snažil navrhovat tak, aby případné budoucí rozšíření mělo co nejmenší vliv na ostatní části a bylo provedeno co nejefektivněji. Nejlépe tedy podle pravidel OOP. Takto vytvořený model se stal součástí aplikační vrstvy aplikace Strip Observer. Současně jsem se v této fázi snažil rozvrhnout potencionální řešení některých algoritmů.

Neméně důležitým úkolem bylo také navržení vhodného rozhraní pro datovou vrstvu. Ačkoliv datová vrstva měla pouze umožňovat číst z různých datových zdrojů, stěžejní bylo navrhnout ji tak, aby případné další rozšíření programu o čtení z nových zdrojů dat bylo co možná nejsnazší.

Jelikož primárním účelem aplikace Strip Observer je provádět teplotní a jiné metalurgické výpočty, bylo nutné vhodným způsobem prezentovat výsledky výpočtů. Proto jako prezentační vrstva byl vytvořen tzv. tlustý klient (fat client), který byl napsán pomocí frameworku MFC. Přičemž o prezentaci výsledků se starala grafická knihovna vytvořená firmou ITA, s. r. o.

Přestože na uživatelské rozhraní aplikace v první iteraci nebyl kladen přílišný důraz, snažil jsem jej vytvořit tak, aby uživatel mohl snadno a intuitivně produkt využívat.

Aplikační vrstvu tvoří staticky linkovaná knihovna. Tato knihovna obsahuje veškerý výkonný kód aplikace. Ve své podstatě se jedná pouze o sadu tříd, které nějakým způsobem implementují požadavky zadavatele. Jelikož jedním z faktorů bylo vytvořit kód, co nejlépe udržovatelný a snadno rozšiřitelný, rozhodl jsem se vytvořit tuto vrstvu jako sadu modulů s co možná nejmenší závislostí na ostatních modulech.

Pod pojmem modul zde myslím jen rozdělení souvisejících tříd do logických části. Tyto části lze v jazyce C++ realizovat pomocí tzv. namespace.

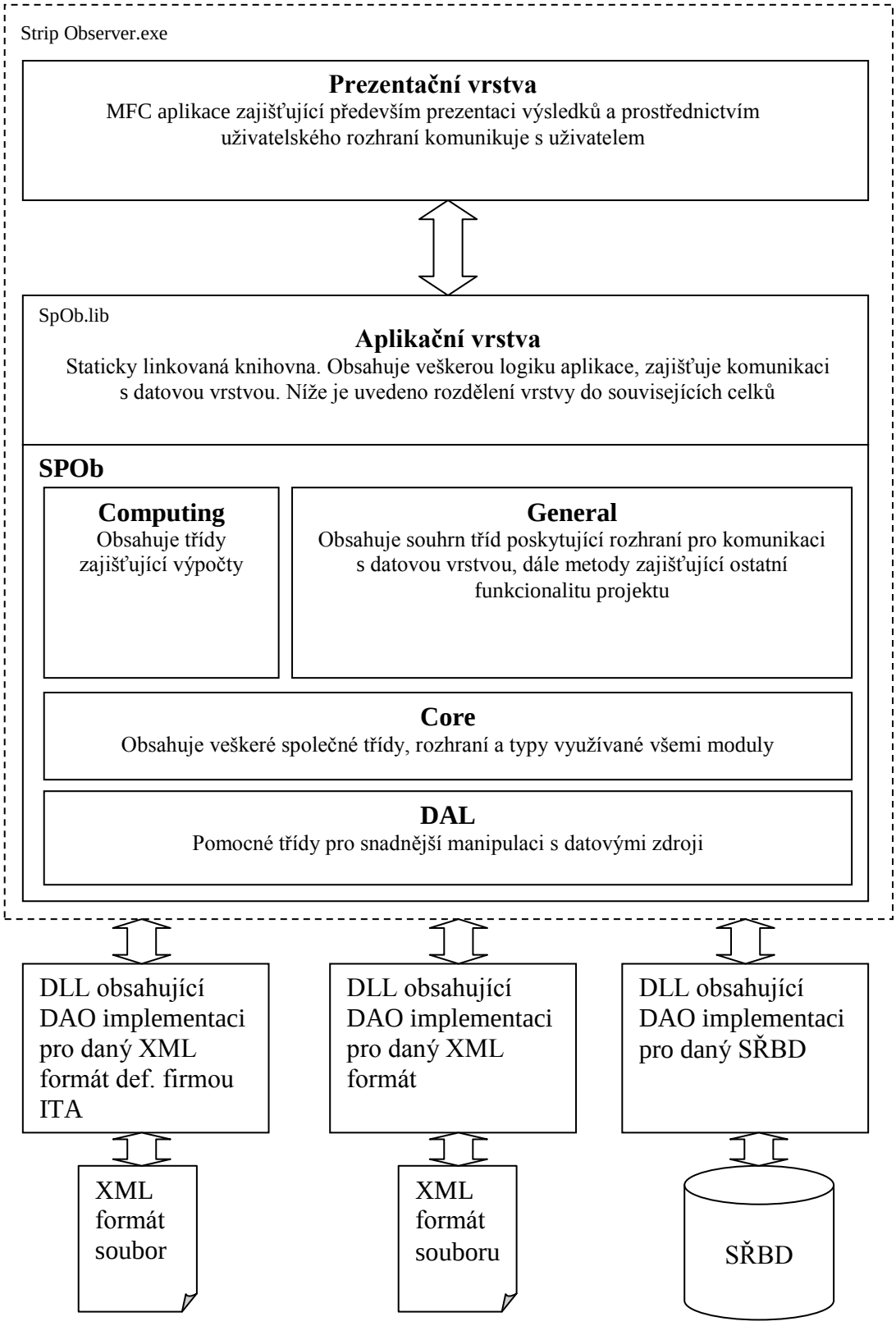
Pro řešení datové vrstvy jsem se kvůli problémům zmíněných v úvodu rozhodl využít mírně upravený návrhový vzor DAO. Po návrhu vhodného rozhraní, které poskytovalo potřebné metody pro komunikaci s aplikací Strip Observer, vyvstávala otázka, jakým způsobem umožnit uživateli za běhu aplikace vyměnit datový zdroj. Shodou okolností existují na platformě Windows v zásadě dva způsoby, jak požadovaného cíle dosáhnout. Jedním z nich je použití dynamicky linkované knihovny (DLL) společně s tzv. explicitním voláním funkcí. Jako další řešení lze použít tzv. COM komponenty.

Přestože se může zdát výhodné použití COM komponent, díky zabudovanému systému volání vzdálených procedur, já si pro program Strip Observer vybral možnost s využitím DLL knihovny. Zejména kvůli jednoduchosti implementace DLL a také díky své neznalosti tvorby COM komponent.

V rámci první verze programu a jeho datové vrstvy byl jako datový zdroj využit XML soubor s předem definovaným formátem. Pro zajištění přístupu k datům uložených v souboru bylo nutné implementovat příslušné datové rozhraní. Skrze tuto implementaci byl program Strip Observer schopen číst data, aniž by musel znát jejich formát.

V závěru tvorby a s blížícím se koncem doby iterace bylo potřeba produkt otestovat. Samotné testování některých částí prováděli mí kolegové. Protože MS Visual Studio 2008 s sebou nepřináší žádné kvalitní nástroje pro testování softwaru napsaného v nativním C++, byl pro testovací účely použit jednoduchý testovací nástroj vytvořený firmou ITA, s. r. o.

Testovací data byla dodána ve formě XML souboru s definovaným formátem. Soubor byl vytvořen exportováním vypočtených dat programem DLPP.



Obrázek 1: Schematické znázornění architektury znázorňující rozdělení aplikační vrstvy do souvisejících modulů

3 UPLATNĚNÉ TEORETICKÉ A PRAKTICKÉ ZNALOSTI

Při své práci na úpravách svěřených projektů jsem ocenil znalost z odborných knih, diskuzí a předmětů Programovací techniky a Programovaní v C++. Při návrhu aplikace mi také velmi pomohly znalosti z předmětu Úvod do softwarového inženýrství, odkud jsem využil především znalosti některých UML diagramů. Dále pak mi byly nápomocné znalosti z předmětu tvorba informačních systémů, kde jsme byli seznámeni s různými architekturami softwaru, a také informace související se značkovacím jazykem XML a jeho technologiemi.

4 SCHÁZEJÍCÍ ZNALOSTI A DOVEDNOSTI

Vzhledem k tomu, že jsem do té doby neměl žádnou zkušenost s tak rozsáhlými projekty, dělaly mi především potíže použité technologie. Rovněž mé znalosti z oblasti stavby aplikací byly spíše teoretické. Nicméně na základě rad svých kolegů a studováním odborných článků jsem postupem času získával potřebné zkušenosti.

5 DOSAŽENÉ VÝSLEDKY A JEJICH ZHODNOCENÍ

Mezi dosažené výsledky bych zařadil úspěšné vytvoření první verze programu Strip Observer, i přestože po předvedení produktu zákazníkovi dosáhl návrh jistých změn. Avšak tyto změny nebyly zvláště zásadní.

Také bych zde zařadil dovednosti a zkušenosti získané během svého působení ve firmě ITA, s. r. o. Díky které jsem dostal jedinečnou možnost pracovat na projektech pro významné zahraniční klienty, stát se součásti týmu a získat tak cenné rady v oblasti vývoje software.

SEZNAM POUŽITÝCH SYMBOLŮ

MFC	Microsft Foundation Classes – Aplikační framework společnosti Microsoft s podporou architektury dokument-pohled.
XML	Extensible Markup Language – Obecný značkovací jazyk.
COM	Component Object Model - .Meziprocesová komunikace vyvinutá firmou Microsoft
DLL	Dynamic Link Library – dynamicky linkovaná knihovna
UML	Unified Modeling Language – sjednocený modelovací jazyk
DAO	Data Acces Object – návrhový vzor pro podporu různých typů datových zdrojů
OOP	Objektově Orientované Programování
MSXML	Microsoft XML – sada služeb pomocí nichž může program pracovat s XML dokumenty
API	Application Programming Interfaces – rozhraní pro programování aplikací

SEZNAM OBRÁZKŮ

SEZNAM PŘÍLOH

bakalářská_práce.doc - kompletní bakalářská práce v DOC formátu

bakalářská_práce.pdf - kompletní bakalářská práce v PDF formátu